**CAPSTONE PROJECT**

**Phase – I**

**Overview**

This test automation project was designed for the Royal Brothers website, a bike rental platform. The goal was to create a robust, scalable, and maintainable test suite using Playwright with TypeScript, focusing on end-to-end functionality, UI validation, and regression testing.

**WEB SITE**: https://www.plumgoodness.com/

**Process Followed**

1. Playwright Configuration

Make sure you have the following installed:

* Node.js v16 or later
* npm comes with Node
* TypeScript support
* Required browsers (Chromium, Firefox, WebKit)

1. Folder Structure

├── tests/ # Test files

├── pages/ # Page Object Models

├── fixtures/ # Custom fixtures and test hooks

├── utils/ # Helpers (e.g., date utils, data loaders)

├── playwright.config.ts # Global test config

└── package.json

1. Page Object Model (POM)

* Created separate files for each page (e.g., HomePage.ts, BookingPage.ts, LoginPage.ts).
* Each page class encapsulates selectors and actions.
* Promotes reusability and easier maintenance.

1. Test Scenarios

Automated key workflows:

* Home Page Load Test: Verified presence of banners, and navigation.
* Search & Booking Flow:
  + Search for bikes by location and dates.
  + Apply filters (brand, engine capacity).
  + Verify availability and pricing.
* User can change multiple locations.
* Validate the Filter functionality and sort by.

1. Fixtures & Hooks

* Created custom fixtures to manage:
  + Browser context per test
  + Reusable login state
* Used beforeAll, beforeEach, and afterEach hooks for setup/cleanup.

1. Playwright Features

* Isolated browser contexts.
* Auto-waiting for elements.
* Cross-browser support Chromium, Firefox, WebKit.
* Screenshot and video capture.
* Custom test fixtures.
* Parallel execution and test sharding.
* Playwright Inspector for debugging.
* Powerful built-in assertions and locators.

1. Continues Integration
   * Integrated with GitHub Actions for automatic test runs.

* Reports are generated with HTML reporter.